(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 11338719 A

(43) Date of publication of application: 10 . 12 . 99

(51) Int. Cl G06F 9/46
G06F 15/16

(21) Application number: 10146768

(22) Date of filing: 28 . 05 . 98

(71) Applicant: FUJITSU LTD

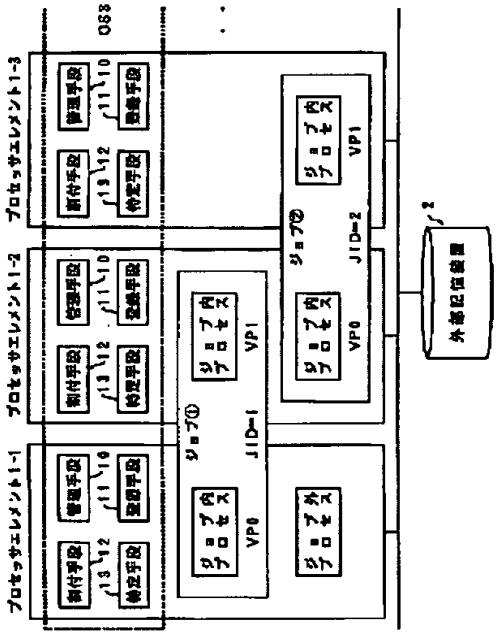
(72) Inventor: YAMASHITA KOICHIRO
WAKABAYASHI HIROHIKO

(54) COMPUTER SYSTEM

(57) Abstract:

PROBLEM TO BE SOLVED: To deal with the save/restoration of a job without reserving any process ID at the time of executing a job provided with plural processor elements and composed of one or plural processes.

SOLUTION: This system is provided with a managing means 10 for managing the relation of correspondence among a job ID allocated to the job, processor ID of a virtual processor element allocated to the job and processor ID of a real processor element for executing the job and an allocating means 12 for generating the process ID defined by coupling the bits of a local process ID defined in the real processor element for executing a process composing of the job when generating that process, virtual processor ID provided for that job instructed by that real processor element and job ID allocated to that job and for allocating the process ID to that process.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平11－338719

(43)公開日　平成11年(1999)12月10日

(51)Int.Cl.⁶		識別記号	FI		
G06F	9/46	360	G06F	9/46	360B
	15/16	380		15/16	380Z

審査請求　未請求　請求項の数3　OL　（全 12 頁）

(21)出願番号　特願平10－146768

(22)出願日　平成10年(1998)5月28日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　山下　浩一郎
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(72)発明者　若林　裕彦
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

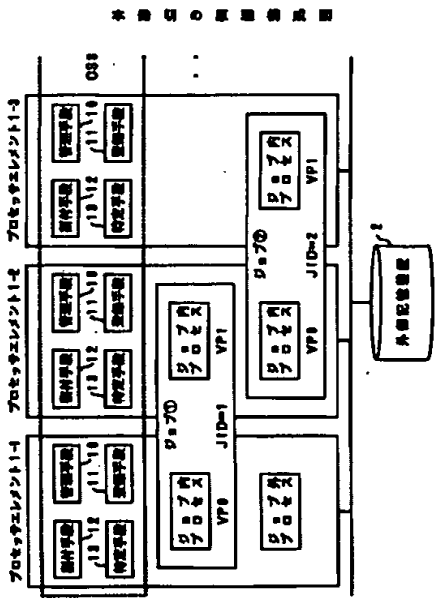
(74)代理人　弁理士　岡田　光由　（外１名）

(54)【発明の名称】　計算機システム

(57)【要約】

【課題】本発明は、複数のプロセッサエレメントを備えて、１つ又は複数のプロセスで構成されるジョブを実行するときにあって、プロセスＩＤを予約することなく、ジョブの退避・復元に対処できるようにすることを目的とする。

【解決手段】ジョブに割り付けられるジョブＩＤと、ジョブに割り付けられる仮想プロセッサエレメントのプロセッサＩＤと、ジョブを実行する実プロセッサエレメントのプロセッサＩＤとの対応関係を管理する管理手段１０と、ジョブを構成するプロセスが生成されるときに、そのプロセスを実行する実プロセッサエレメント内で定義されるローカルプロセスＩＤと、その実プロセッサエレメントの指すそのジョブの持つ仮想プロセッサＩＤと、そのジョブに割り付けられるジョブＩＤとのビット結合で定義されるプロセスＩＤを生成して、そのプロセスに割り付ける割付手段１２とを備えるように構成する。



1

【特許請求の範囲】

【請求項１】　複数のプロセッサエレメントを備えて、１つ又は複数のプロセスで構成されるジョブを実行する計算機システムにおいて、

ジョブに割り付けられるジョブＩＤと、ジョブに割り付けられる仮想プロセッサエレメントのプロセッサＩＤと、ジョブを実行する実プロセッサエレメントのプロセッサＩＤとの対応関係を管理する管理手段と、

ジョブを構成するプロセスが生成されるときに、該プロセスを実行する実プロセッサエレメント内で定義されるローカルプロセスＩＤと、該実プロセッサエレメントの指す該ジョブの持つ上記仮想プロセッサＩＤと、該ジョブに割り付けられるジョブＩＤとのビット結合で定義されるプロセスＩＤを生成して、該プロセスに割り付ける割付手段とを備えることを、

特徴とする計算機システム。

【請求項２】　請求項１記載の計算機システムにおいて、

割付手段は、ジョブを構成しないプロセスが生成されるときに、該プロセスを実行する実プロセッサエレメント内で定義されるローカルプロセスＩＤと、該実プロセッサエレメントのプロセッサＩＤとのビット結合で定義されるプロセスＩＤを生成して、該プロセスに割り付けることを、

特徴とする計算機システム。

【請求項３】　請求項１又は２記載の計算機システムにおいて、

プロセスＩＤを指定して通信要求が発行されるときに、該プロセスＩＤと管理手段の管理データとから通信先となるプロセスを特定する特定手段を備えることを、

特徴とする計算機システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数のプロセッサエレメントを備えて、１つ又は複数のプロセスで構成されるジョブを実行する計算機システムに関し、特に、プロセスＩＤを予約することなく、ジョブの退避・復元に対処できるようにする計算機システムに関する。

【０００２】

【従来の技術】計算機システムでは、互いのジョブの唯一性を保証するために、ジョブに対して、互いにユニークなジョブＩＤを割り付ける。

【０００３】一方、並列に拡張された計算機システムでは、ジョブは１つ又は複数のプロセスで構成されており、ジョブに属するプロセスに対しても、ジョブに属さないプロセスに対しても、互いにユニークなプロセスＩＤを割り付ける。

【０００４】複数のプロセッサエレメントで構成される並列プロセッサシステムでは、ジョブＩＤとプロセスＩＤとはプロセッサエレメントを跨がって唯一性を保証しなければならない。

2

【０００５】このような条件の下、プロセス群から構成されるジョブを外部記憶装置に退避した後に、他の操作を行うと、退避したジョブに属するプロセス群に割り付けられているプロセスＩＤが、後で生成されたプロセスのプロセスＩＤと重なることが起こり、これから、退避したジョブを復元する場合にプロセスＩＤの衝突が発生することがある。

【０００６】このプロセスＩＤの衝突は、退避したプロセスが多い場合や、退避中に頻繁にプロセスの生成・終了が行われる場合に、その発生の確率が高くなる。このプロセスＩＤの衝突を回避するために、従来では、使用の予約されているプロセスＩＤを登録する予約表を用意する構成を採って、プロセスを外部記憶装置に退避するときに、そのプロセスに割り付けられているプロセスＩＤを予約表に登録することで、プロセスの退避中に、そのプロセスのプロセスＩＤが後から生成されるプロセスに割り付けられないようにするという構成を採っていた。

【０００７】

【発明が解決しようとする課題】しかしながら、このような従来技術に従っていると、予約表に登録されたプロセスＩＤについては使用できないことから、後から生成されるプロセスに割り付けられるプロセスＩＤの採番範囲が圧迫されるという問題点があった。

【０００８】更に、退避するプロセスの数に応じたメモリ容量を持つ予約表を用意する必要があるという問題点があった。本発明はかかる事情に鑑みてなされたものであって、複数のプロセッサエレメントを備えて、１つ又は複数のプロセスで構成されるジョブを実行する構成を採るときにあって、プロセスＩＤを予約することなく、ジョブの退避・復元に対処できるようにする新たな計算機システムの提供を目的とする。

【０００９】

【課題を解決するための手段】図１に本発明を具備する計算機システムの原理構成を図示する。この図に示すように、本発明を具備する計算機システムは、複数のプロセッサエレメント１-ｉ（ｉ＝１,2，・・）と外部記憶装置２とで構成されて、オペレーティングシステム３に、管理手段１０と、登録手段１１と、割付手段１２と、特定手段１３とを備える構成を採る。

【００１０】この管理手段１０は、ジョブ（１つ又は複数のプロセスで構成される）に割り付けられるジョブＩＤと、ジョブに割り付けられる仮想プロセッサエレメントのプロセッサＩＤと、ジョブを実行するプロセッサエレメント１-ｉの実プロセッサＩＤとの対応関係を管理する。

【００１１】登録手段１１は、ジョブが生成されるときに、管理手段１０の管理するデータを生成して、それを管理手段１０に登録する。割付手段１２は、プロセスが

生成されるときに、そのプロセスの識別子となるプロセスＩＤを生成して、それをそのプロセスに割り付ける。

【００１２】特定手段１３は、プロセスＩＤを指定して通信要求が発行されるときに、通信先となるプロセスを特定する。このように構成される本発明を具備する計算機システムでは、登録手段１１の登録処理に従って、管理手段１０は、ジョブＩＤと仮想プロセッサＩＤと実プロセッサＩＤとの対応関係を管理する。

【００１３】例えば、管理手段１０は、図１のプロセッサエレメント１-1,2に展開されるジョブ①については、図２に示すように、“１”というジョブＩＤ（ＪＩＤ）を持ち、“ＶＰ０”と“ＶＰ１”という２つの仮想プロセッサＩＤを持ち、実プロセッサＩＤとして、仮想プロセッサＩＤの“ＶＰ０”に割り付けられるプロセッサエレメント１-1のＰＥｘと、仮想プロセッサＩＤの“ＶＰ１”に割り付けられるプロセッサエレメント１-2のＰＥｙとを持つということを管理する。

【００１４】また、管理手段１０は、図１のプロセッサエレメント１-2,3に展開されるジョブ②については、図２に示すように、“２”というジョブＩＤ（ＪＩＤ）を持ち、“ＶＰ０”と“ＶＰ１”という２つの仮想プロセッサＩＤを持ち、実プロセッサＩＤとして、仮想プロセッサＩＤの“ＶＰ０”に割り付けられるプロセッサエレメント１-2のＰＥｙと、仮想プロセッサＩＤの“ＶＰ１”に割り付けられるプロセッサエレメント１-3のＰＥｚとを持つということを管理する。

【００１５】一方、割付手段１２は、ジョブを構成するプロセスが生成されると、そのプロセスを実行するプロセッサエレメント１-i内で定義されるローカルプロセスＩＤと、そのプロセッサエレメント１-iの指すそのジョブの持つ仮想プロセッサＩＤと、そのジョブのジョブＩＤとのビット結合で定義される図３（ａ）に示すようなプロセスＩＤを生成して、それをそのプロセスに割り付ける。

【００１６】また、割付手段１２は、ジョブを構成しないプロセスが生成されると、そのプロセスを実行するプロセッサエレメント１-i内で定義されるローカルプロセスＩＤと、そのプロセッサエレメント１-iの実プロセッサＩＤとのビット結合で定義される図３（ｂ）に示すようなプロセスＩＤを生成して、それをそのプロセスに割り付ける。

【００１７】ここで、図３中に示す「符号」は、ジョブを構成するプロセスであるのか否かを示す識別符号である。この本発明のプロセスＩＤを使うことで、ジョブを外部記憶装置２に退避した後に、他の操作を行うことで、後で生成されたプロセスに対して、退避したジョブの持つプロセスに割り付けられているものと同一のローカルプロセスＩＤが割り付けられるようなことが起きても、何も不都合は起こらない。

【００１８】また、この本発明のプロセスＩＤを使うことで、ジョブをプロセッサエレメント１-i間で移動するときに、移動先に、そのジョブの持つプロセスに割り付けられているものと同一のローカルプロセスＩＤを持つプロセスが展開されているようなことがあっても、何も不都合は起こらない。

【００１９】すなわち、本発明のプロセスＩＤは、従来技術で使われているローカルプロセスＩＤを拡張したデータ構造を持つので、この拡張部分のデータ構造の違いにより、同一のローカルプロセスＩＤを持つようなことがあってもプロセスを識別することができるのである。

【００２０】この本発明のプロセスＩＤを用いるときに、プロセスＩＤを指定して通信要求が発行されると、特定手段１３は、そのプロセスＩＤの持つ「符号」がジョブを構成しないプロセスであることを示しているときには、そのプロセスＩＤの持つ実プロセッサＩＤの指すプロセッサエレメント１-iに展開される、そのプロセスＩＤの持つローカルプロセスＩＤの指すプロセスを通信先として特定する。

【００２１】一方、特定手段１３は、通信要求先として指定されるプロセスＩＤの持つ「符号」がジョブを構成するプロセスであることを示しているときには、そのプロセスＩＤの持つジョブＩＤ／仮想プロセッサＩＤを検索キーにして管理手段１０を検索することで実プロセッサＩＤを特定して、その特定した実プロセッサＩＤの指すプロセッサエレメント１-iに展開される、そのプロセスＩＤの持つローカルプロセスＩＤの指すプロセスを通信先として特定する。

【００２２】このようにして、本発明のプロセスＩＤを用いるときには、管理手段１０の管理データを参照することで、簡単に通信要求先のプロセスを特定できるようになる。

【００２３】

【発明の実施の形態】以下、実施の形態に従って本発明を詳細に説明する。図４に本発明を具備する計算機システムの一実施例を図示する。

【００２４】この実施例に従う本発明を具備する計算機システムは、ＰＥｘ、ＰＥｙ、ＰＥｚ、ＰＥαという４台のプロセッサエレメントと、外部記憶装置２とを備えている。

【００２５】各プロセッサエレメント上では、オペレーティングシステムが動作しており、システムの運用のために、単一あるいは複数のプロセス群から構成される複数のジョブと、ジョブを構成しない複数のプロセスとが動作している。

【００２６】各プロセッサエレメントには、オペレーティングシステムから通知されるジョブ管理表１００が展開される。このジョブ管理表１００は、ジョブに割り付けられるジョブＩＤと、ジョブに割り付けられる仮想プロセッサＩＤと、ジョブを実行するプロセッサエレメントの実プロセッサＩＤとの対応関係を管理する。

【００２７】オペレーティングシステムは、ジョブがシステムに投入されると、そのジョブに対してユニークなジョブＩＤを割り付けるとともに、そのジョブを実行するプロセッサエレメントに対して、相対的な通番で定義される仮想プロセッサＩＤを割り付け、それを各プロセッサエレメントに展開されるジョブ管理表１００に登録する。ここで、ジョブＩＤと仮想プロセッサＩＤとは、そのジョブが存在している間は不変値として扱われる。

【００２８】図４の実施例で説明するならば、システムに投入されるジョブ①に対して、“１”というジョブＩＤを割り付けるとともに、そのジョブ①を実行するプロセッサエレメントＰＥｘに対して“ＶＰ０”という仮想プロセッサＩＤを割り付け、そのジョブ①を実行するプロセッサエレメントＰＥｙに対して“ＶＰ１”という仮想プロセッサＩＤを割り付けて、それらをジョブ管理表１００に登録するのである。

【００２９】また、システムに投入されるジョブ②に対して、“２”というジョブＩＤを割り付けるとともに、そのジョブ②を実行するプロセッサエレメントＰＥｙに対して“ＶＰ０”という仮想プロセッサＩＤを割り付け、そのジョブ②を実行するプロセッサエレメントＰＥｚに対して“ＶＰ１”という仮想プロセッサＩＤを割り付けて、それらをジョブ管理表１００に登録するのである。

【００３０】そして、オペレーティングシステムは、ジョブが操作されると、その操作に応じて、必要に応じてジョブ管理表１００の管理データを書き換えていくよう処理する。

【００３１】例えば、図４に示したジョブ②が図５に示すように移動されることで、ジョブ②を実行するプロセッサエレメントがＰＥｙ／ＰＥｚからＰＥｚ／ＰＥαに移動すると、ジョブ②を実行するプロセッサエレメントＰＥｚに対して“ＶＰ０”という仮想プロセッサＩＤを割り付け、ジョブ②を実行するプロセッサエレメントＰＥαに対して“ＶＰ１”という仮想プロセッサＩＤを割り付けて、それらに従ってジョブ管理表１００の管理データを書き換えていくのである。

【００３２】なお、このジョブ管理表１００は、標準のＵＮＩＸシステムのシェルが持つジョブの管理表を配列変数を用いて並列に拡張したものである。更に、オペレーティングシステムは、プロセスが生成されると、図６の処理フローに従って、そのプロセスに対して、本発明に特徴的なプロセスＩＤを割り付ける。

【００３３】次に、図６に示す処理フローに従って、このプロセスＩＤの割付処理について説明する。オペレーティングシステムは、プロセスが生成されると、図６の処理フローに示すように、先ず最初に、ステップ１で、生成されたジョブに対して、単一プロセッサ構成を採る標準のＵＮＩＸシステムで用いられる１６ビットのプロセスＩＤ（以下、ローカルプロセスＩＤと称する）を生成する。

【００３４】続いて、ステップ２で、生成されたプロセスがジョブを構成するものであるのか否かを判断して、ジョブを構成することを判断するときには、ステップ３に進んで、生成されたプロセスの属するジョブのジョブＩＤを特定し、続くステップ４で、生成されたプロセスを実行するプロセッサエレメントの実プロセッサＩＤを特定する。

【００３５】続いて、ステップ５で、ステップ３で特定したジョブＩＤと、ステップ４で特定した実プロセッサＩＤとを検索キーにして、ジョブ管理表１００を検索することで、生成されたプロセスに対応付けられる仮想プロセッサＩＤを特定する。

【００３６】続いて、ステップ６で、ステップ１で生成したローカルプロセスＩＤと、ステップ３で特定したジョブＩＤと、ステップ５で特定した仮想プロセッサＩＤとをビット結合するとともに、その先頭にジョブを構成するプロセスのプロセスＩＤであることを示すフラグ値“１”（ジョブ判定フラグ）を付加することで、図７（ａ）に示すような３２ビットのプロセスＩＤを生成して、それを生成されたプロセスに割り付ける。

【００３７】一方、ステップ２で、生成されたプロセスがジョブを構成しないことを判断するときには、ステップ７に進んで、生成されたプロセスを実行するプロセッサエレメントの実プロセッサＩＤを特定する。

【００３８】続いて、ステップ８で、ステップ１で生成したローカルプロセスＩＤと、ステップ７で特定した実プロセッサＩＤとをビット結合するとともに、その先頭にジョブを構成しないプロセスのプロセスＩＤであることを示すフラグ値“０”（ジョブ判定フラグ）を付加することで、図７（ｂ）に示すような３２ビットのプロセスＩＤを生成して、それを生成されたプロセスに割り付ける。

【００３９】このようにして、単一プロセッサ構成を採る標準のＵＮＩＸシステムでは、プロセスに対して、図７（ｃ）に示すような１６ビットのプロセスＩＤ（本発明に言うローカルプロセスＩＤ）を割り付ける構成を採るのに対して、本発明を具備する計算機システムでは、並列プロセッサシステムを前提とするときに、ＵＮＩＸシステムの豊富な資産を引き継ぐための互換性を考慮して、下位２バイトに、ＵＮＩＸシステムで割り付けるローカルプロセスＩＤを持つとともに、上位２バイトに、ジョブに属するプロセスのときには、ジョブ判定フラグ／ジョブＩＤ／仮想プロセッサＩＤを持ち、ジョブに属さないプロセスのときには、ジョブ判定フラグ／実プロセッサＩＤを持つプロセスＩＤを割り付ける構成を採るのである。

【００４０】このようにして、例えば、図４に示すジョブ①の持つプロセッサエレメントＰＥｘに展開されるプロセスに対して、

〔1，1，ＶＰ0，ａａａａ〕
但し、“ａａａａ”はローカルプロセスＩＤ
という３２ビットのプロセスＩＤが割り付けられ、図４に示すジョブ①の持つプロセッサエレメントＰＥｙに展開されるプロセスに対して、
〔1，1，ＶＰ1，ｂｂｂｂ〕
但し、“ｂｂｂｂ”はローカルプロセスＩＤ
という３２ビットのプロセスＩＤが割り付けられ、図４に示すプロセッサエレメントＰＥｘに展開されるジョブ外プロセスに対して、
〔0，ＰＥｘ，ｃｃｃｃ〕
但し、“ｃｃｃｃ”はローカルプロセスＩＤ
という３２ビットのプロセスＩＤが割り付けられることになる。

【００４１】このプロセスＩＤを使うことで、本発明を具備する計算機システムでは、ジョブを外部記憶装置２に退避した後に、他の操作を行うことで、後で生成されたプロセスに対して、退避したジョブの持つプロセスに割り付けられているものと同一のローカルプロセスＩＤが割り付けられるようなことが起きても、何も不都合は起こらない。

【００４２】また、このプロセスＩＤを使うことで、本発明を具備する計算機システムでは、ジョブをプロセッサエレメント間で移動するときに、移動先に、そのジョブの持つプロセスに割り付けられているものと同一のローカルプロセスＩＤを持つプロセスが展開されているようなことがあっても、何も不都合は起こらない。

【００４３】すなわち、本発明を具備する計算機システムで用いるプロセスＩＤは、標準のＵＮＩＸシステムで使われているローカルプロセスＩＤを拡張したデータ構造を持つので、この拡張部分のデータ構造の違いにより、同一のローカルプロセスＩＤを持つようなことがあってもプロセスを識別することができるのである。

【００４４】オペレーティングシステムは、ジョブ内プロセス又はジョブ外プロセスから、プロセスＩＤを指定して通信要求が発行されると、図８の処理フローに従って、通信要求先のプロセスを特定する処理を行う。

【００４５】次に、この通信要求先の特定処理について説明する。オペレーティングシステムは、プロセスＩＤを指定して通信要求が発行されると、図８の処理フローに示すように、先ず最初に、ステップ１で、指定されるプロセスＩＤの先頭の１ビットを読み取る。

【００４６】続いて、ステップ２で、読み取ったビット値から，指定されるプロセスＩＤの指すプロセスがジョブ内プロセスであるのか、ジョブ外プロセスであるのかをチェックして、ジョブ内プロセスであることを判断するときには、ステップ３に進んで、ビット演算による切出処理を実行することで、指定されるプロセスＩＤから、ジョブＩＤ／仮想プロセッサＩＤ／ローカルプロセスＩＤを切り出す。

【００４７】続いて、ステップ４で、切り出したジョブＩＤ及び仮想プロセッサＩＤを検索キーとしてジョブ管理表１００を検索することで、実プロセッサＩＤを特定し、続くステップ５で、特定した実プロセッサＩＤの指すプロセッサエレメントに展開される、ステップ３で切り出したローカルプロセスＩＤの指すプロセスを通信先として特定する。

【００４８】このようにして、通信要求先として指定されるプロセスＩＤの指すプロセスがジョブ内プロセスであるときには、図９に示すように、指定されるプロセスＩＤとジョブ管理表１００の管理データとから通信先を特定するのである。

【００４９】一方、ステップ２で、読み取ったビット値から、指定されるプロセスＩＤの指すプロセスがジョブ外プロセスであることを判断するときには、ステップ６に進んで、ビット演算による切出処理を実行することで、指定されるプロセスＩＤから、実プロセッサＩＤ／ローカルプロセスＩＤを切り出す。

【００５０】続いて、ステップ７で、切り出した実プロセッサＩＤの指すプロセッサエレメントに展開される、切り出したローカルプロセスＩＤの指すプロセスを通信先として特定する。

【００５１】このようにして、通信要求先として指定されるプロセスＩＤの指すプロセスがジョブ外プロセスであるときには、図１０に示すように、指定されるプロセスＩＤから通信先を特定するのである。

【００５２】通信要求先として指定されるプロセスＩＤの指すプロセスがジョブ内プロセスであるときの通信先の特定処理は、５τのマシンサイクルで実行できるので、高速に通信先を特定することが可能である。

【００５３】すなわち、この通信先の特定処理は、先頭の１ビットのビット値（ジョブ判定フラグ）を参照することで、指定されるプロセスＩＤがジョブに属するのかを判断して、ジョブに属することを判断するときには、ジョブ管理表１００を検索することで実行されることになるが、先頭の１ビットのビット値が“１”か“０”であるのかは、３τで実行される比較処理により実行され、ジョブ管理表１００の検索は、２τで実行される２次元配列の指定処理により実行されるので、合計５τのマシンサイクルで実行できるのである。

【００５４】上述したように、本発明を具備する計算機システムで用いるプロセスＩＤは、単一プロセッサ構成を採る標準のＵＮＩＸシステムで用いられるプロセスＩＤを拡張したものとなっている。

【００５５】従って、標準のＵＮＩＸシステムの豊富な資産を引き継ぐことが可能となる。更に、単一プロセッサ構成を採る標準のＵＮＩＸシステムから、本発明を具備する並列プロセッサ構成の計算機システムへのマイグレード操作（移動操作）が可能になるとともに、本発明を具備する並列プロセッサ構成の計算機システムから、

単一プロセッサ構成を採る標準のＵＮＩＸシステムへのマイグレード操作（移動操作）が可能になるという特徴を有する。

【００５６】すなわち、本発明を具備する計算機システムにプロセッサエレメントが追加されるときに、既存のＵＮＩＸシステムで動作するプロセスのプロセスＩＤを拡張（図７に示すように上位２バイトを付加する）することで、その既存のＵＮＩＸシステムで動作するプロセスを本発明を具備する計算機システムに移動させることが可能になる。これにより、システムを簡単に拡張できるようになる。

【００５７】また、本発明を具備する計算機システムのプロセッサエレメントの１つがダウンするときに、そのプロセッサエレメントで動作するプロセスのプロセスＩＤの上位２バイトを削除（移動先が無視するときには削除する必要はない）することで、そのプロセッサエレメントで動作するプロセスを標準のＵＮＩＸシステムにに移動させることが可能になる。これにより、システムダウンに簡単に対処できるようになる。

【００５８】図１１に、本発明を具備する計算機システムで実行する運用処理の処理フロー、図１２に、本発明を具備する計算機システムと連携する、単一プロセッサ構成を採る標準のＵＮＩＸシステムで実行する運用処理の処理フローを図示する。

【００５９】次に、この処理フローについて説明する。本発明を具備する計算機システムは、起動されると、図１１の処理フローで示す処理に入って、ジョブ（プロセス）に対する操作に従って、ユーザによりジョブ（プロセス）が投入されると、ローカルプロセスＩＤを採番し、ジョブ判定フラグ／ジョブＩＤ／仮想プロセッサＩＤや、ジョブ判定フラグ／実プロセッサＩＤという属性情報を適宜配置し結合し、それらとローカルプロセスＩＤとを結合することで、投入されるジョブ（プロセス）に対して、３２ビットのプロセスＩＤを割り付けていく。

【００６０】一方、ジョブ（プロセス）に対する操作に従って、ユーザによりシグナルが発行されたり、プロセス間の通信指示が発行されると、ビット演算により属性情報を抽出し、プロセスＩＤから処理対象のプロセスを特定して、それに対する制御や処理を行う。

【００６１】一方、ジョブ（プロセス）に対する操作に従って、外部記憶装置２への退避や外部記憶装置２からの復元が行われると、外部記憶装置２への退避のときには、ユーザの指示に従ってジョブ（プロセス）を外部記憶装置２に退避させた後、ユーザの指示に従って他ジョブを投入してシステムを再起動する。また、外部記憶装置２からの復元のときには、ユーザからの指示が非移動復元（元のプロセッサエレメントに戻す復元）であるのか移動復元であるのかを判断して、非移動復元のときには、ビット演算により属性情報を抽出し、プロセスＩＤから処理対象のプロセスを特定して、それに対する制御や処理を行い、移動復元のときには、移動先が自システムであるのか標準のＵＮＩＸシステムであるのかを判断して、移動先が自システムのときには、ビット演算により属性情報を抽出し、プロセスＩＤから処理対象のプロセスを特定して、それに対する制御や処理を行う。

【００６２】一方、ジョブ（プロセス）に対する操作に従って、ユーザによりマイグレード操作（移動操作）が行われると、移動先が自システムであるのか標準のＵＮＩＸシステムであるのかを判断して、移動先が自システムのときには、ビット演算により属性情報を抽出し、プロセスＩＤから処理対象のプロセスを特定して、それに対する制御や処理を行う。

【００６３】この処理構成を採るときに、移動先が標準のＵＮＩＸシステムであるときには、図１１及び図１２のＡに示すように、移動対象となるジョブ（プロセス）を標準のＵＮＩＸシステムへ移動させる。このとき、標準のＵＮＩＸシステムでは、３２ビットのプロセスＩＤの内、下位１６ビットのプロセスＩＤ部分のみをプロセスＩＤとして抽出するので、移動対象となるジョブ（プロセス）をそのまま標準のＵＮＩＸシステムへ移動させていくことになる。

【００６４】また、本発明を具備する計算機システムと連携するＵＮＩＸシステムは、起動されると、図１２の処理フローに示す処理に入って、ジョブ（プロセス）に対する操作に従って、ユーザによりジョブ（プロセス）が投入されると、図７（ｃ）に示した１６ビットのプロセスＩＤを採番することで、投入されるジョブ（プロセス）に対して、１６ビットのプロセスＩＤを割り付けていく。

【００６５】一方、ジョブ（プロセス）に対する操作に従って、ユーザによりシグナルが発行されたり、プロセス間の通信指示が発行されると、１６ビットのプロセスＩＤから処理対象のプロセスを特定して、それに対する制御や処理を行う。

【００６６】一方、ジョブ（プロセス）に対する操作に従って、ユーザによりマイグレード操作（移動操作）が行われると、図１１及び図１２のＢに示すように、移動対象となるジョブ（プロセス）を本発明を具備する計算機システムへ移動させ、そのとき、１６ビットのジョブ判定フラグ／ジョブＩＤ／仮想プロセッサＩＤや、ジョブ判定フラグ／実プロセッサＩＤという属性情報を適宜配置し結合し、それらとＵＮＩＸシステムで付加された１６ビットのプロセスＩＤとを結合することで、移動されるジョブ（プロセス）に対して、３２ビットのプロセスＩＤを割り付けていく。

【００６７】このようにして、本発明を具備する計算機システムは、単一プロセッサ構成を採る標準のＵＮＩＸシステムの豊富な資産を引き継ぎながら、データ処理を行うことができるのである。

11

【００６８】この実施例では、ジョブに属するプロセスとジョブに属さないプロセスとを区別するという構成を採ったが、ジョブに属さないプロセスについても、ジョブを構成するものと見なす方法を採ることも可能である。この方法を採る場合には、プロセスに割り付けられるプロセスＩＤは、図７（ａ）に示すものに統一されることになる。

【００６９】

【発明の効果】以上説明したように、本発明によれば、並列プロセッサ構成を採る計算機システムにあって、標準のＵＮＩＸシステムの豊富な資産をそのまま引き継ぎながら、プロセスＩＤを予約することなく、ジョブの退避・復元に対処できるようになる。

【図面の簡単な説明】

【図１】本発明の原理構成図である。

【図２】管理手段の管理するデータの説明図である。

【図３】本発明のプロセスＩＤの説明図である。

12

【図４】本発明の一実施例である。

【図５】本発明の一実施例である。

【図６】プロセスＩＤの割付処理の処理フローである。

【図７】本発明のプロセスＩＤの一実施例である。

【図８】通信先の特定処理の処理フローである。

【図９】通信先の特定処理の説明図である。

【図１０】通信先の特定処理の説明図である。

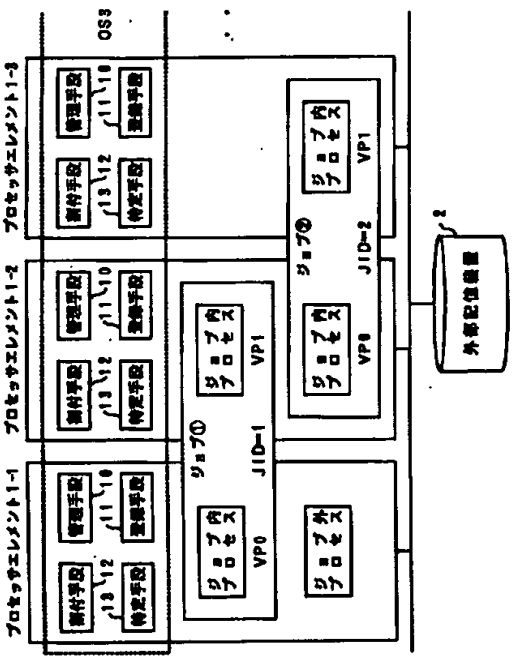
【図１１】本発明の運用処理の説明図である。

【図１２】本発明の運用処理の説明図である。

【符号の説明】

１　プロセッサエレメント
２　外部記憶装置
３　オペレーティングシステム
１０　管理手段
１１　登録手段
１２　割付手段
１３　特定手段

【図１】

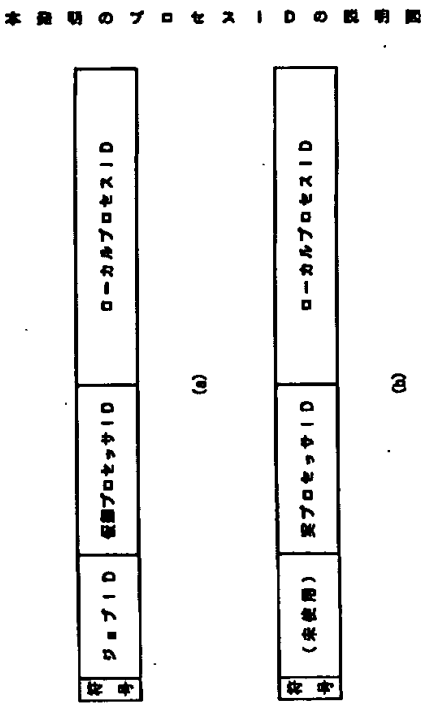


【図２】

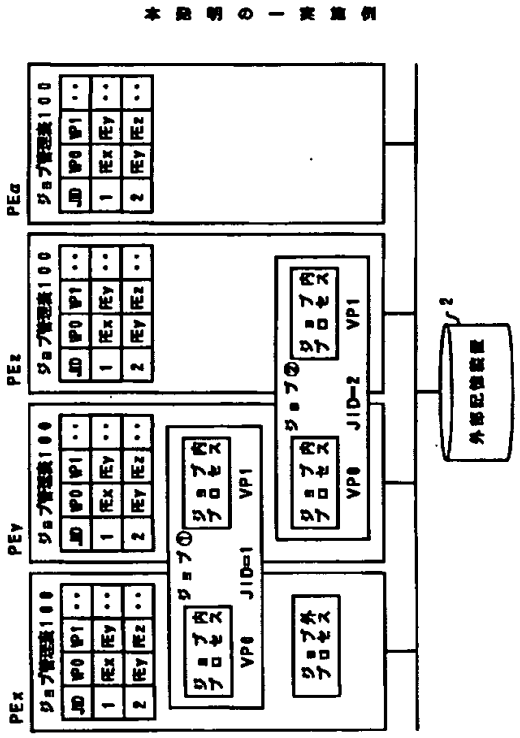
管理手段の管理するデータの説明図

JID	VP0	VP1	VP2	・・・・
1	PEx	PEy	—	・・・・
2	PEy	PEz	—	・・・・
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・
・	・	・	・	・

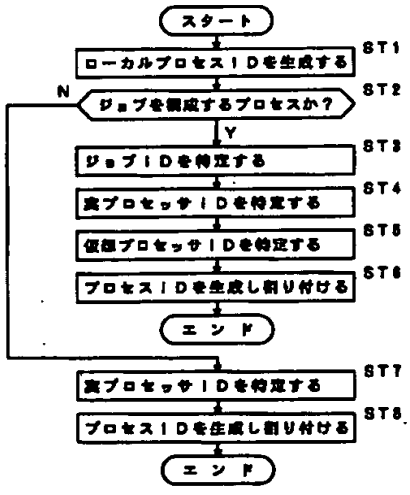
【図3】


本発明のプロセスＩＤの説明図
ローカルプロセスＩＤ
仮想プロセッサＩＤ
ジョブＩＤ
符号
(a)
ローカルプロセスＩＤ
実プロセッサＩＤ
（未使用）
符号
(b)

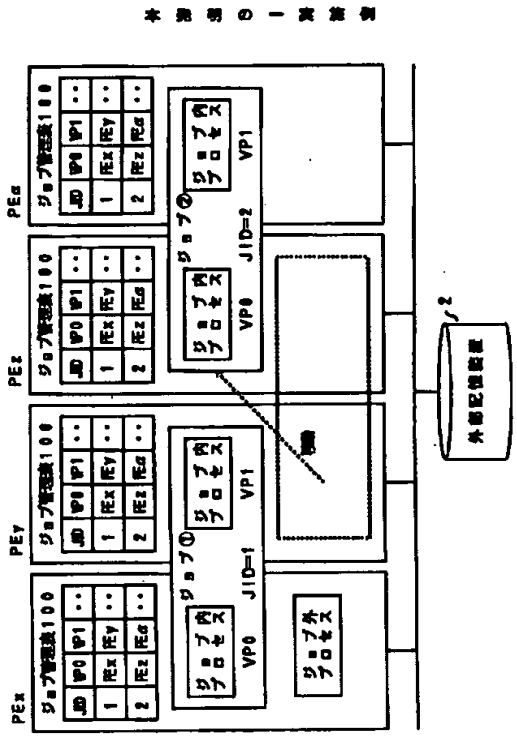
【図4】


本発明の一実施例
PE0
PE1
PEy
PEx
ジョブ管理テーブル100
JID
VP0
VP1
PE0
PE1
1
2
ジョブ内プロセス
ジョブ外プロセス
ジョブ①
ジョブ②
JID=1
JID=2
外部記憶装置
2

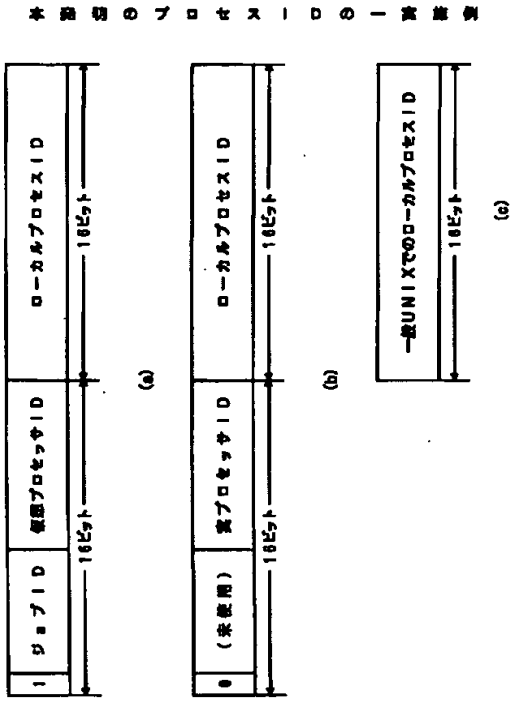
【図6】


プロセスＩＤの割付処理の処理フロー
スタート
ローカルプロセスＩＤを生成する
ST1
ジョブを構成するプロセスか？
ST2
N
Y
ジョブＩＤを特定する
ST3
実プロセッサＩＤを特定する
ST4
仮想プロセッサＩＤを特定する
ST5
プロセスＩＤを生成し割り付ける
ST6
エンド
実プロセッサＩＤを特定する
ST7
プロセスＩＤを生成し割り付ける
ST8
エンド

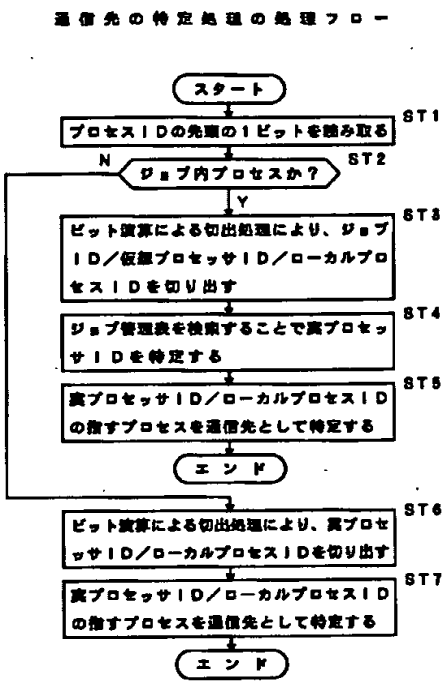
【図５】


本発明の一実施例
PEx
PEy
PEz
PEa
ジョブ管理表100
JID
VP0
VP1
REx
REy
REz
REa
0
1
2
ジョブ①
ジョブ②
ジョブ内プロセス
ジョブ外プロセス
JID=1
JID=2
VP0
VP1
共有記憶装置
2

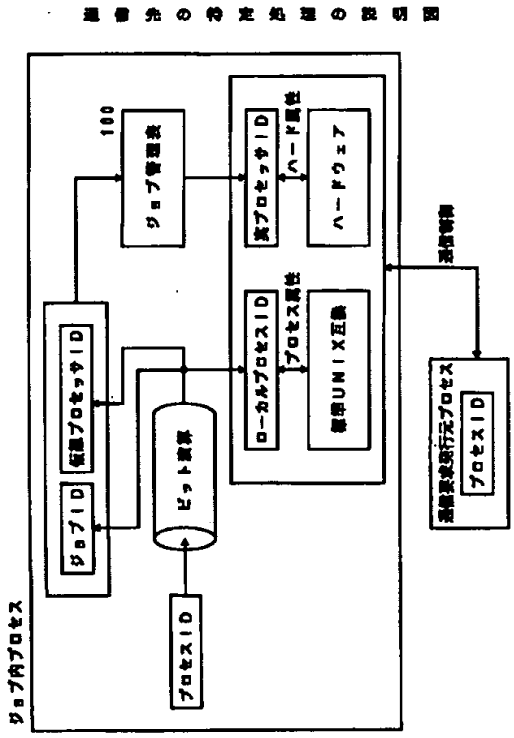
【図７】


本発明のプロセスIDの一実施例
ジョブID
物理プロセッサID
ローカルプロセスID
(未使用)
実プロセッサID
ローカルプロセスID
実UNIXでのローカルプロセスID
16ビット
(a)
(b)
(c)

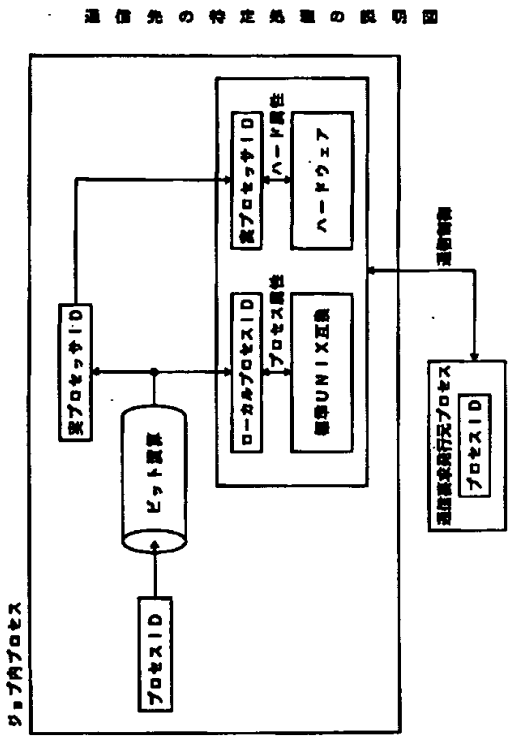
【図８】


通信先の特定処理の処理フロー
スタート
プロセスＩＤの先頭の１ビットを読み取る
ST1
ジョブ内プロセスか？
ST2
N
Y
ビット演算による切出処理により、ジョブＩＤ／仮想プロセッサＩＤ／ローカルプロセスＩＤを切り出す
ST3
ジョブ管理表を検索することで実プロセッサＩＤを特定する
ST4
実プロセッサＩＤ／ローカルプロセスＩＤの指すプロセスを通信先として特定する
ST5
エンド
ビット演算による切出処理により、実プロセッサＩＤ／ローカルプロセスＩＤを切り出す
ST6
実プロセッサＩＤ／ローカルプロセスＩＤの指すプロセスを通信先として特定する
ST7
エンド

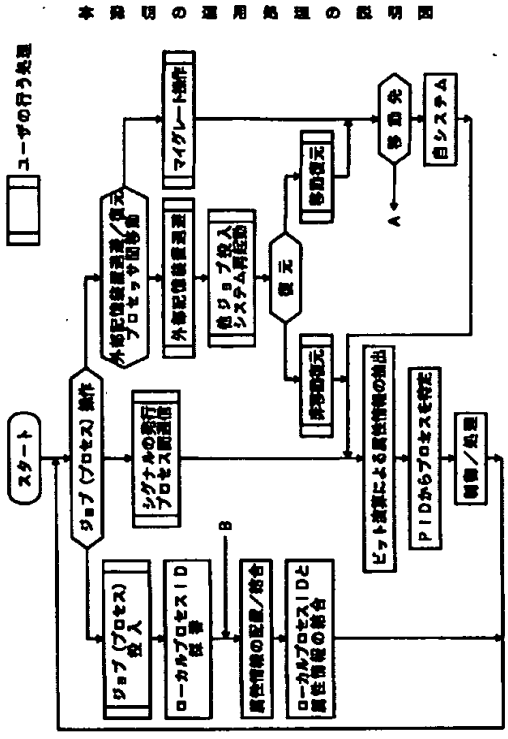
【図９】


通信先の特定処理の説明図
100
ジョブ管理表
実プロセッサＩＤ
ハードウェア
ハード属性
プロセス属性
ローカルプロセスＩＤ
標準ＵＮＩＸ互換
通信情報
通信要求発行元プロセス
プロセスＩＤ
仮想プロセッサＩＤ
ジョブＩＤ
ビット演算
プロセスＩＤ
ジョブ内プロセス

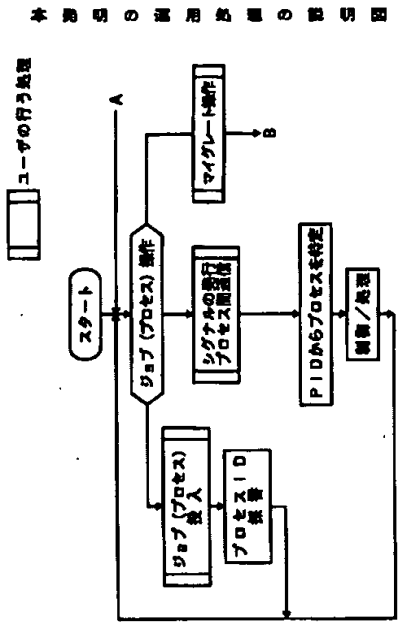
【図１０】


通信先の特定処理の説明図
ジョブ内プロセス
プロセスID
ビット演算
要プロセッサID
ローカルプロセスID
プロセス属性
標準UNIX互換
要プロセッサID
ハード属性
ハードウェア
通信情報
通信実行元プロセス
プロセスID

【図１１】


本処理の運用処理の説明図
ユーザの行う処理
スタート
ジョブ（プロセス）操作
シグナルの発行プロセス属性指定
外部記憶装置読込／復元プロセッサ切り替え
外部記憶装置読込
他ジョブ投入システム再起動
復元
マイグレート操作
移動復元
移動先
A
自システム
終了処理復元
ビット演算による属性情報の抽出
PIDからプロセスを特定
制御／処理
B
ジョブ（プロセス）投入
ローカルプロセスID作成
属性情報の記憶／統合
ローカルプロセスIDと属性情報の結合

【図１２】


本発明の適用処理の説明図
ユーザの行う処理
スタート
A
ジョブ（プロセス）操作
マイグレーション操作
B
シグナルの発行プロセス指定
PIDからプロセス名特定
結果／表示
ジョブ（プロセス）登録
プロセスID検索